Teaching Physical Based Animation via OpenGL Slides

Miao Song
Graduate School,
Concordia University,
Montreal, Canada,
Email: m_song@cse.concordia.ca

Serguei A. Mokhov
Computer Science
and Software Engineering,
Concordia University, Montreal, Canada,
Email: mokhov@cse.concordia.ca

Peter Grogono
Computer Science
and Software Engineering,
Concordia University, Montreal, Canada,
Email: grogono@cse.concordia.ca

Abstract

This work expands further our earlier poster presentation and integration of the OpenGL Slides Framework (OGLSF) – to make presentations with real-time animated graphics where each slide is a scene with tidgets – and physical based animation of elastic two-, three-layer softbody objects. The whole project is very interactive, and serves dual purpose – delivering the teaching material in a classroom setting with real running animated examples as well as releasing the source code to the students to show how the actual working things are made.

Keywords: education, presentation, softbody, real-time, frameworks, OpenGL, physical-based modeling

1 Introduction

It is very helpful for effective teaching of computer graphics (CG) techniques [20, 3, 19], especially advanced topics such as real-time physical based animation of softbody objects, with a real working code on hand that can be demonstrated in a classroom and then given out to students for learning purposes and extension for their course work. It is reasonable to assume, in subjects like CG, teaching may be less effective if the examples are not visualized in class for the students. On top of that, it can be a nuisance for the instructor presenting the concepts and switching between the presentation power-point-like slides and the demo especially if it is complex and highly interactive with a lot of variable parameters to tweak. As a result, for the cases like the one briefly described, we argue that it is more effective *and* efficient to combine the OpenGL CG programs with OpenGL presentation slides in one teaching unit. There the traditional power-points and various techniques can be exemplified at run-time at the same time and the source code can be released to the students later to follow the examples through at all angles. For this purpose we integrated the physical-based softbody simulation system [11, 12, 13] with the OpenGL slides presentation framework (OGLSF) [6, 7] that compose in a small demo that we discuss throughout this work.

Organization

In Section 2 we discuss the background and the related work done that contribute to the creation of this teaching unit, specifically we discuss the properties of the OGLSF in Section 2.1 and the Softbody Simulation System in Section 2.2. We then describe a brief methodology and layout in Section 3. Afterwards, we conclude in Section 4 describing our achievement, the limitation of the approach in Section 4.1, and

the future work items in Section 4.2. All the sections are illustrated with the actual screenshots from the said OpenGL softbody system presentation slides and referenced where appropriate.

2 Related Work

The major pieces of the related work that contribute to this works, are two frameworks alongside with their implementation, put together with other items, to eventually form a teaching module for computer graphics. Here we extrapolate from our previous poster presentation [17] on this topic with more details on the actual design and implementation of the physical-based softbody animation techniques in an OpenGL power-point-like presentation tool with the demonstrated results. We describe the two CG systems in this section in some detail for the unaware reader.

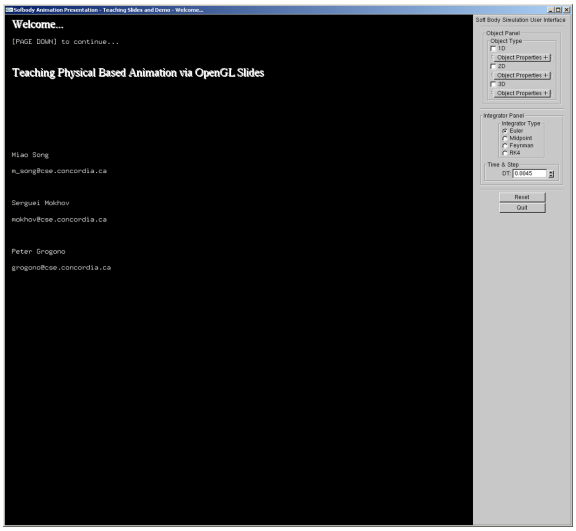
2.1 OpenGL Slides Framework (OGLSF)

OGLSF gives ability to make slides, navigate between them using various controls, and allow for common bulleted textual widgets – the *tidgets*. It also allows to override the control handling from the main idle loop down to each individual (current) slide. All slides together compose a concrete instance of `Presentation`, which is a collection of slides that uses the Builder pattern to sequence the slides. Each slide is a derivative of the generic `Slide` class and represents a scene with the default keyboard controls for the tidgets and navigation. It is understood that the tidgets can be enabled and disabled to allow the main animation to run unobstructed [6, 7, 17]. Each scene on the slide is modeled using traditional procedural modeling techniques [21] and is set as a developer or artist desires. It can include models and rendering of any primitives, complex scenes, texturing, lighting, GPU-based shading, and others, as needed and is fit by the presenter [6, 7]. The main program delegates its handling of the callback controls for keyboard, mouse, and idle all the way down to the presentation object that handles it and passes it down to each current slide [6, 7, 17].

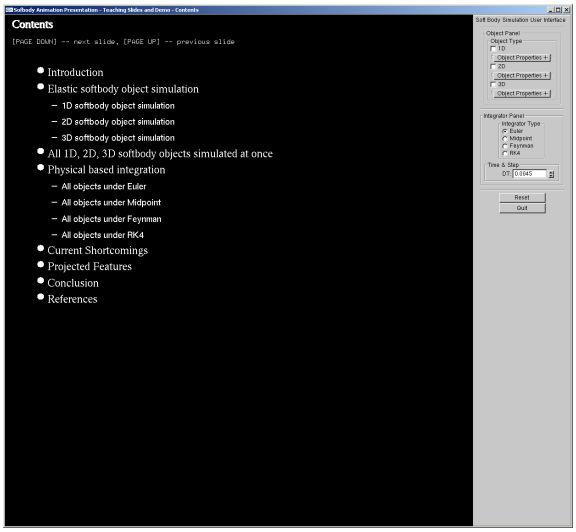
2.2 Softbody Simulation Framework and System

The Softbody Simulation System's main goal is to provide real-time simulation of a variety of softbody objects, founded in the core two- or three- layer model for objects such as human and animal's soft parts and tissue, and non-living soft objects, such as cloth, gel, liquid, and gas. Softbody simulation is a vast research topic and has a long history in computer graphics. The softbody system has gone through a number of iterations in its design and development. Initially it had limited user interface [11, 12]. Then the fine-grained to high-level level-of-detail (LOD) GLUI-based [9] user interface has been added [13], GPU shading support was added [14] using the OpenGL Shading Language [10], a curve-based animation was integrated, and software engineering re-design is constantly being applied. The example of the common visual design of the LOD interactivity interface is summarized in Figure 2. The LOD components are on the right-hand-side, expanded, and the main simulation window is on the left (interactivity with that window constitutes for now just the mouse drag and functional keys). Following the top-down approach configuration parameters, that assume some defaults, were reflected in the GUI [17].

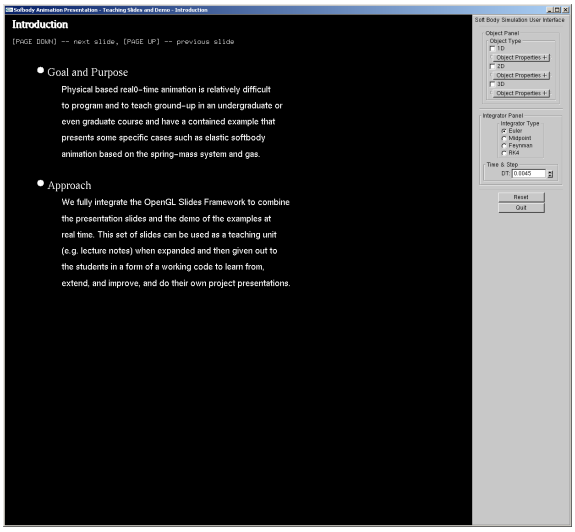
The core framework's design is centered around common dimensionality (1D, 2D, and 3D) of graphical objects for simulation purposes, physics-based integrators, and the user interaction component. The `Integrator` API of the framework as of this writing is implemented by the well-known Explicit Euler, Midpoint, Feynman, and Runge-Kutta 4 (RK4)-based integrators for their mutual comparison of the run-time and accuracy. The system is implemented using OpenGL [8, 22] and the C++ programming language with the object oriented programming paradigm [17].



(a) Welcome Title Slide



(b) Table of Contents Slide



(c) Introduction Slide

Figure 1: A Set of Introductory Tidget-only Slides

This elastic object simulation system has been designed and implemented according to the well known architectural pattern, the model-view-controller (MVC). This pattern is ideal for real-time simulation because it simplifies the dynamic tasks handling by separating data (the model) from user interface (the view). Thus, the user's interaction with the software does not impact the data handling; the data can be reorganized without changing the user interface. The communication between the model and the view is done through the controller. This also closely correlates to the OpenGL state machine, that is used as a core library for the implementation [12, 17].

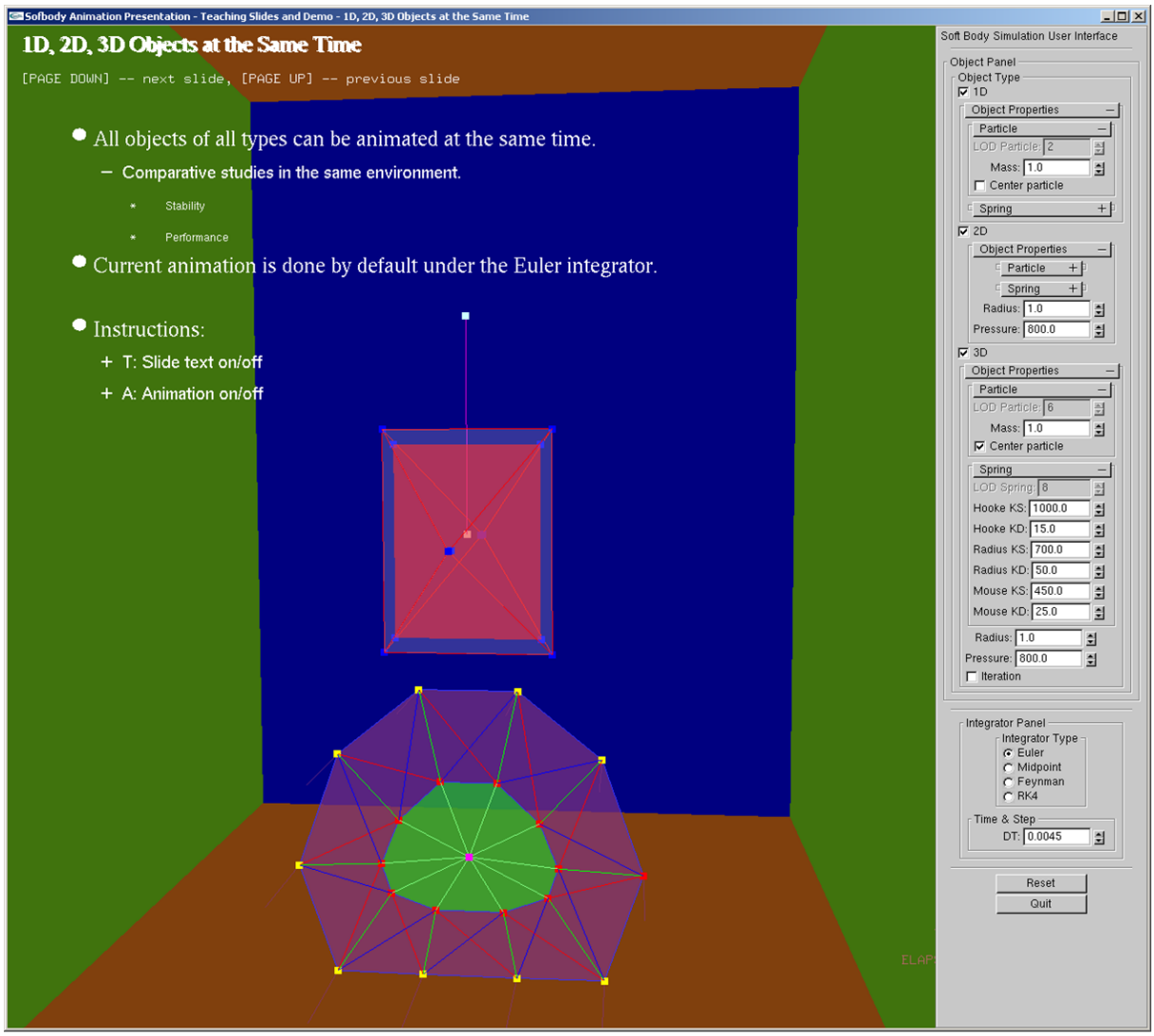
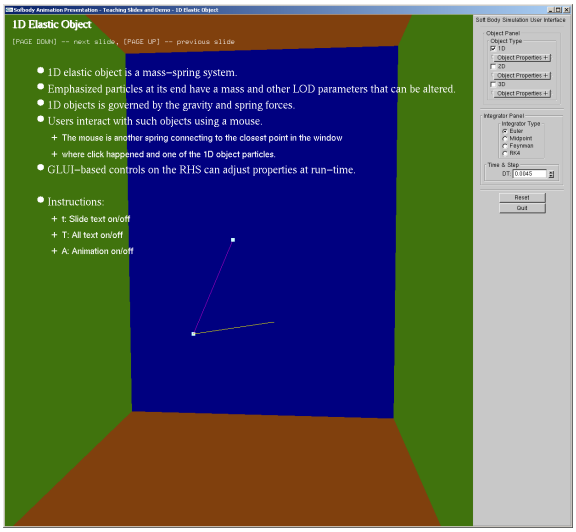


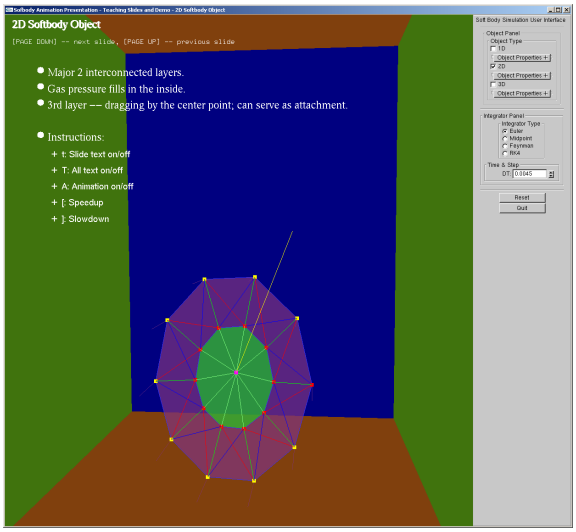
Figure 2: Three Types of Softbody Objects Simulated Together on a Single Slide Scene

3 Methodology

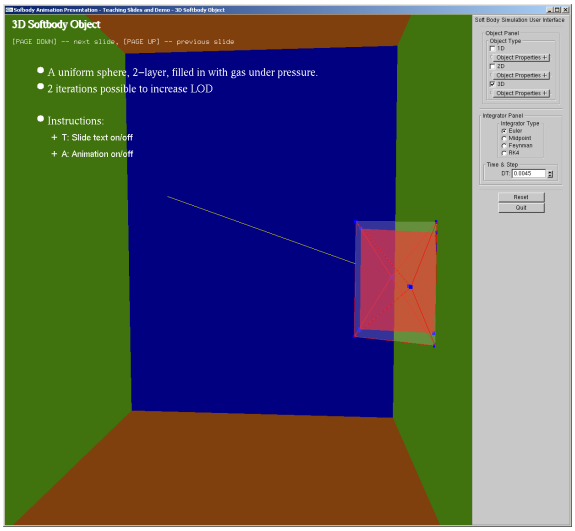
The methodology consists of the design and implementation modification required for the integration followed by making the actual presentation slides. The source code of the presentation is a part of the learning material along the actual content of the material presented and is prepared as such. Separately, both frameworks and implementing systems define the `main()` function, which cannot be included into any of the libraries (both can be compiled into the library files to be linked into other projects) because of the linker errors when the object code from the two or more systems is combined into a single executable. We therefore started a new application with a new `main()`, the `SoftBodyPresentation.cpp`. Additionally, both frameworks have to declare their own namespaces, which both have not done in the past, similarly to CUGL [2] because there are some common names of variables, classes, or functions that clash on compilation. This is an overall improvement not only for this work, but also for any similar type of integration with other projects (cf. Section 4.2). Thus, we declared the namespaces `softbody:`



(a) 1D Elastic Object Simulation Slide



(b) 2D Elastic Object Simulation Slide



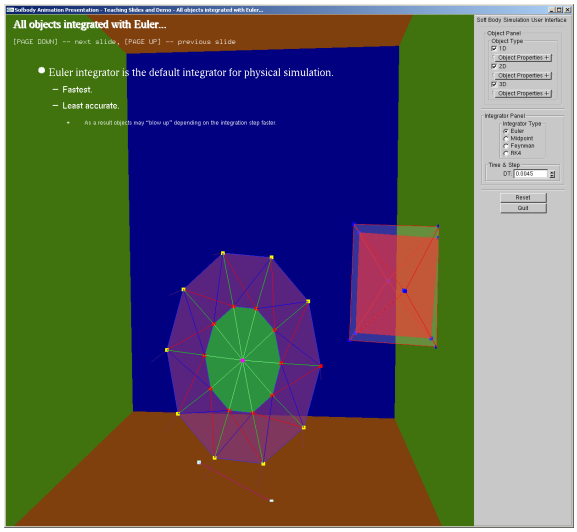
(c) 3D Elastic Object Simulation Slide

Figure 3: Simulation of Single 1D, 2D, and 3D Softbody Elastic Objects Slides

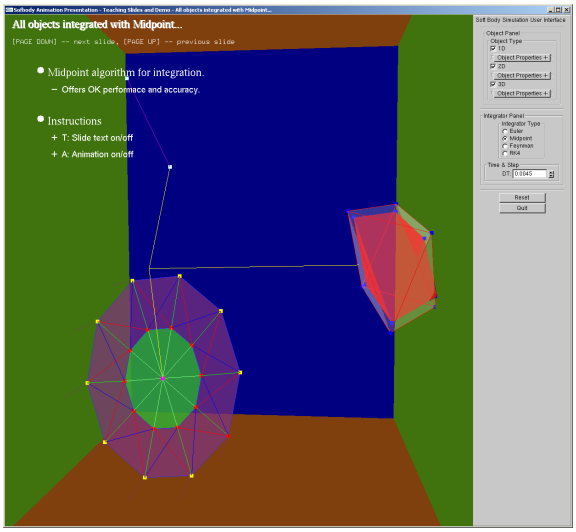
and `slides:` and move the clashing variables under those namespaces. Most of the main code from `SoftbodySimulation.cpp` application is encapsulated into a generic `SoftbodySimulationSlide` class that includes the default configuration of the softbody simulation parameters [17] this class is inherited by the slides that do the actual simulation of softbody objects. Furthermore, the concrete slides that inherit from `SoftbodySimulationSlide` are broken down into some preset distinct configuration defaults and accompanying tidgets. They override the `animate()` method (the "idle" function) as well as the state LOD parameters per an example slide.

For the presentation in this work the demonstration slides currently include the following:

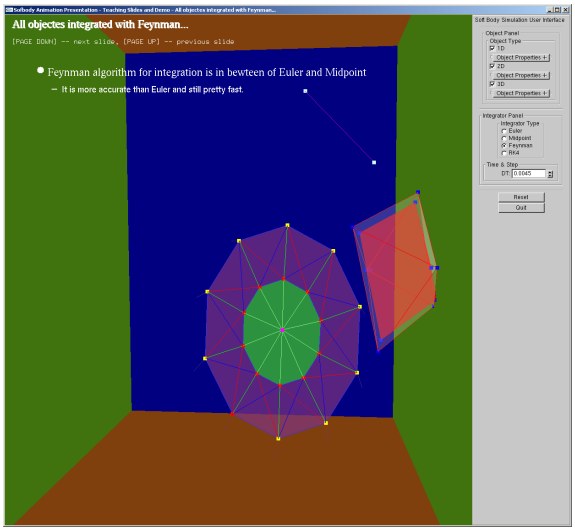
1. `TitleSlide` – a typical title slide with the lecture/presentation title and the presenter information (see Figure 1(a)).



(a) Simulation with Euler Integrator Slide



(b) Simulation with Midpoint Integrator Slide



(c) Simulation with Feynman Integrator Slide

Figure 4: Simulation of all Softbody Object Types with Various Integrators Slides

2. `TOCSlide` – a tidget table of contents of the presentation (see Figure 1(b)).
3. `IntroductionSlide` – a tidget introduction of the material (see Figure 1(c)).
4. `SoftbodySimulationSlide1D` – a slide featuring the 1D elastic object configured by default encased in the `ViewBox`, see Figure 3(a).
5. `SoftbodySimulationSlide2D` – a slide featuring the 2D softbody object configured by default, see Figure 3(b).
6. `SoftbodySimulationSlide3D` – a slide featuring the 3D softbody object configured by default, see Figure 3(c).

7. `SoftbodySimulationSlideAllD` – a slide featuring all types of softbody objects configured by default, as shown in Figure 2, included into the slide environment.

8. `SoftbodySimulationSlideAllEuler` – all three objects configured by default to animate under the Explicit Euler integrator, see Figure 4(a).

9. `SoftbodySimulationSlideAllMidpoint` – all three objects configured by default to animate under the Midpoint integrator, see Figure 4(b).

10. `SoftbodySimulationSlideAllFeynman` – all three objects configured by default to animate under the Feynman integrator, see Figure 4(c).

11. `SoftbodySimulationSlideAllRK4` – all three objects configured by default to animate under the RK4 integrator, see Figure 5.

12. `ShortcomingsSlide` – a slide describing the limitations of the approach (cf. Section 4.1, see Figure 7(a)).

13. `ProjectedFeaturesSlide` – a summary of some projected features for the future work (cf. Section 4.2, see Figure 7(b)).

14. `ConclusionSlide` – a preliminary conclusions slide (cf. Section 4, see Figure 6).

15. `ReferencesSlide` – the list of references, see Figure 7(c).

4 Conclusions and Future Work

We completed the first proof-of-concept integration of the softbody simulation system and OGLSF frameworks. We made a number of slides in a OpenGL-based softbody presentation typically found in lab/tutorial like presentations, which are to be extended to a full lecture-type set of slides. This milestone significantly advances our contribution to a good CG teaching module, suitable for use by instructors to present the material in class as well as for learning by providing its source code to the students for study and extension to demonstrate their CG projects at the end of a semester.

We have encountered some integration difficulties due to the frameworks' original design and implementation consideration, that we do not discuss here, but rather discuss and generalize at length in our follow up software engineering work in [16].

We further discuss the limitation of the proposed approach as well as the future (and ongoing) work on these and the related projects.

4.1 Limitations

There are some assumptions and limitations to the approach described here; thus, it is not an all-in-one solution, but rather for a specific purpose presentations.

- Assumes the CG topics taught are renderable at real-time, like this softbody simulation.
  - One can presumably also render images that were premade offline, but then what is the point? (Unless one is to demonstrate the texture mapping, etc. of course).
  - One can play AVI and HD movies in OpenGL, but again not as a primary learning source, though may be necessary at times (e.g. to teach how to play such things when/if needed).

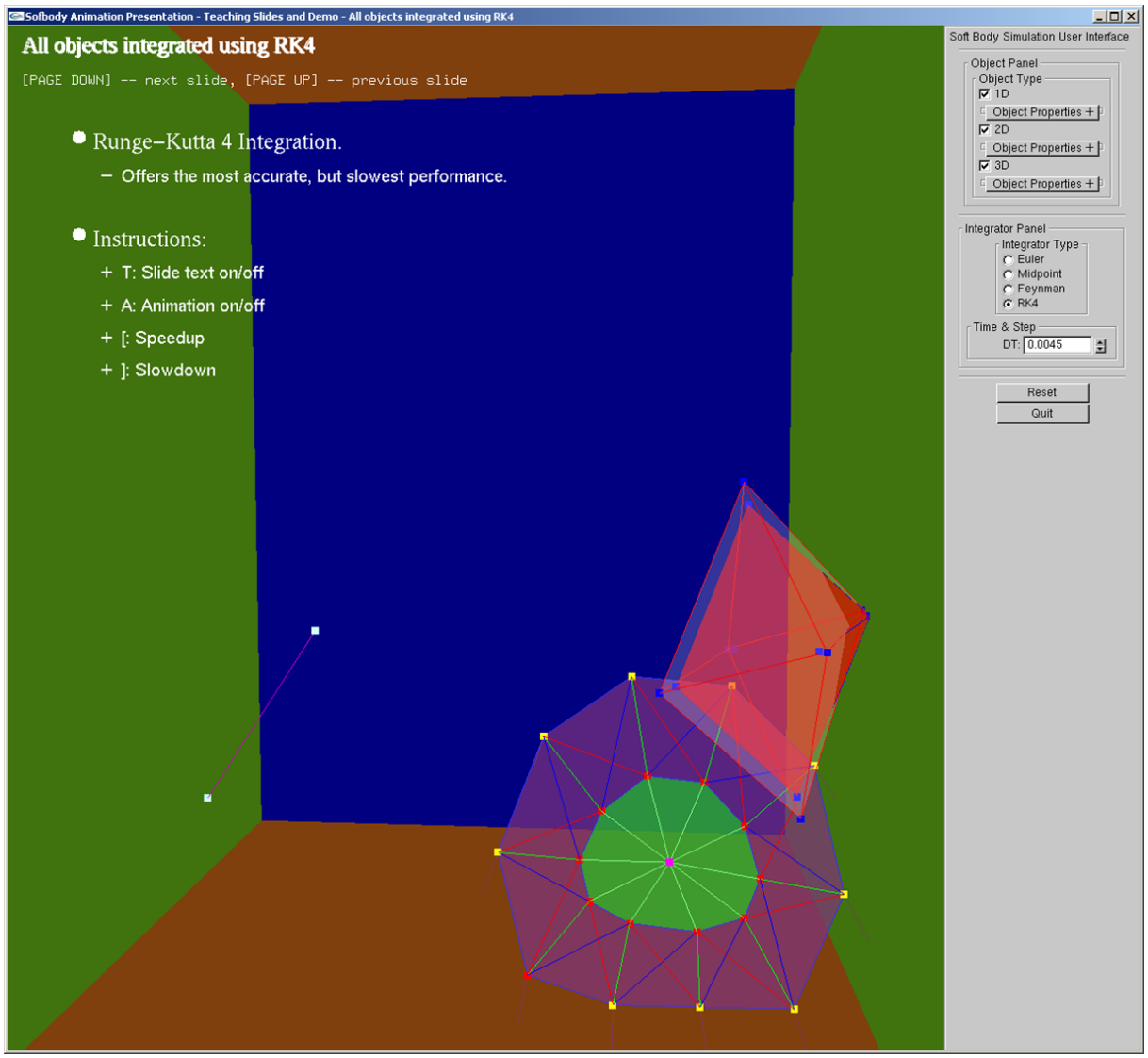


Figure 5: Simulation with Runge-Kutta 4 Integrator Slide

- Not suitable for presentation in online conferences, so have to make screenshots (not a big deal, but ideally, at least the screenshots should be taken automatically).
- May be hardware dependent.
  - Though today's commodity hardware should generally be good enough, but one own's laptop and the PC in a classroom may have enough differences to distort the presentation or render it unacceptably.
- Tedious to compose and debug – need to be a programmer.
  - A proposal to load import info from XML or text is made.
  - Has to be done well in advance before the teaching session.

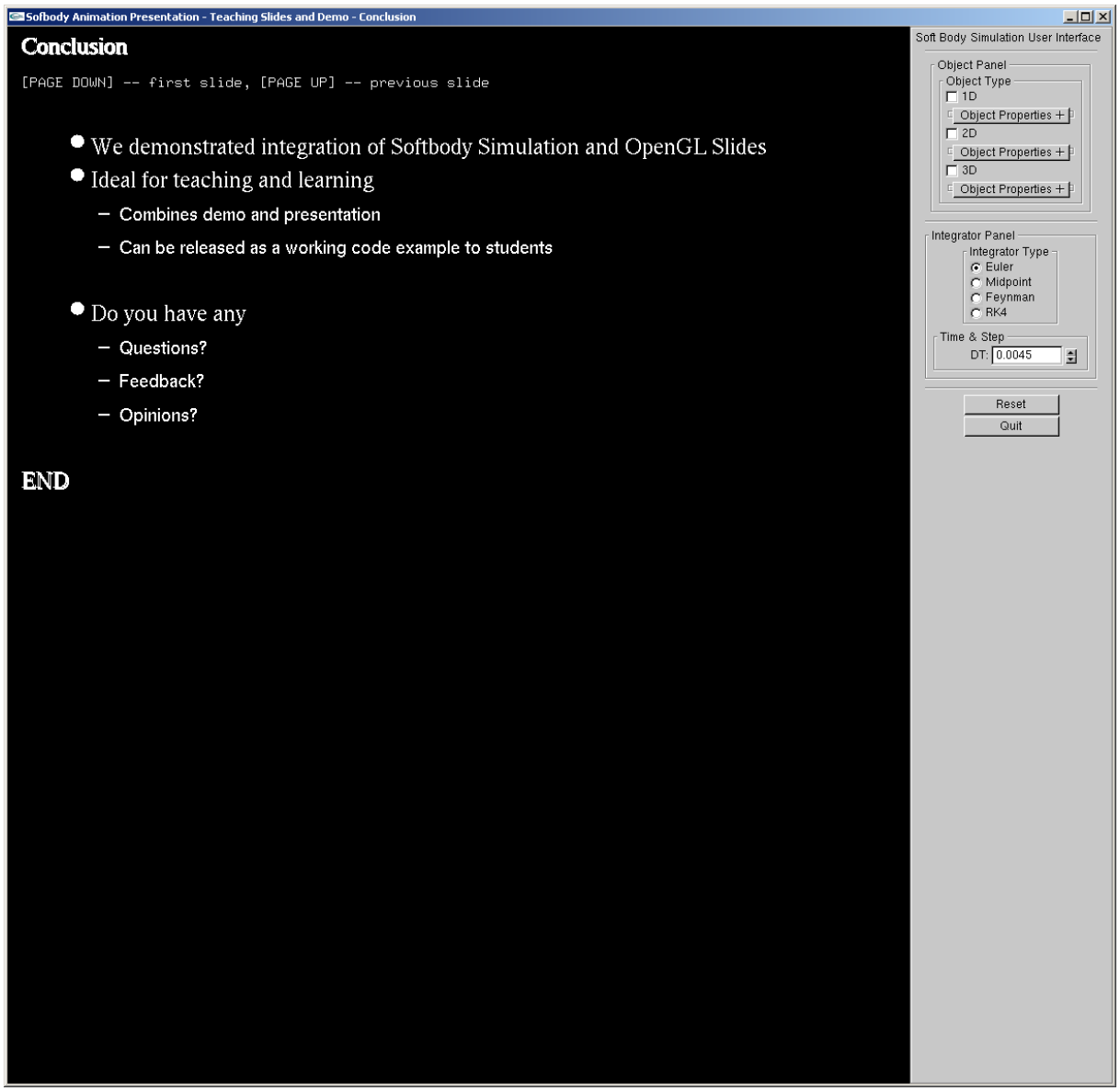
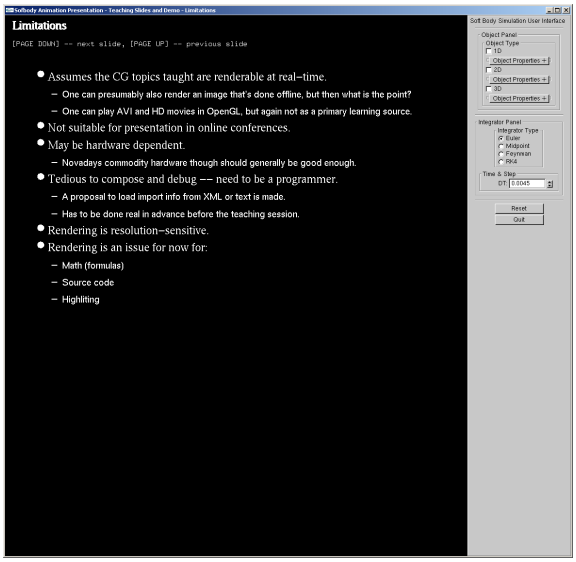


Figure 6: Conclusion OpenGL Slide Example

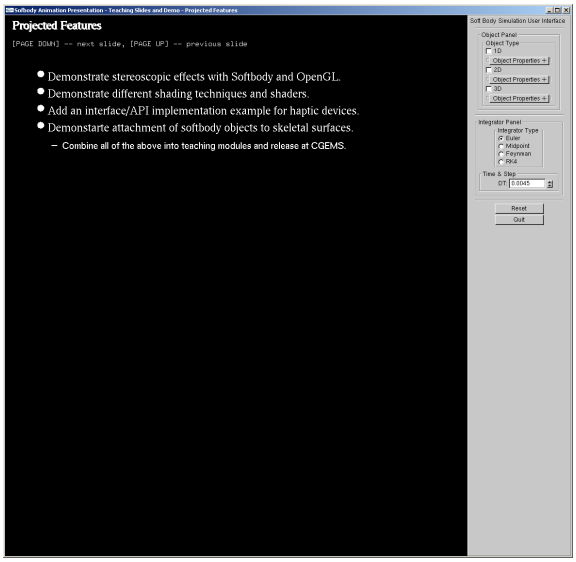
  - Assumption is made the instructors and the students are able to do C++ or OpenGL programming in a CG programming course.

- Rendering is resolution-sensitive – tidgets may have various spacing or stretch, not matching the softbody object being displayed in the scene.

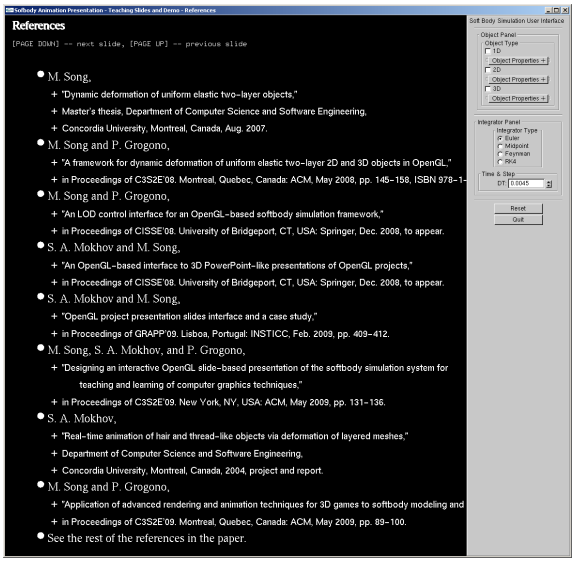
- Rendering is an issue (not supported, or tedious/difficult) for now for math (formulas), source code, highlighting – can only be done as images.



(a) Limitations OpenGL Slide Example



(b) Projected Features OpenGL Slide Example



(c) References OpenGL Slide Example

Figure 7: Concluding Slides

4.2 Future Work

Aside from addressing some of the limitations from the previous section, there are a number of immediate items for the future work:

- Port the source code fully to Linux and MAC OS X. Currently it only compiles properly under WINDOWS XP 32-BIT under Visual Studio 2005.
- Release our code and documentation as open-source implementation either a part of the Concordia University Graphics Library [2] and/or as part of a Maya [1] plug-in and as a CGEMS [3] teaching

module.

- Allow advanced interactive controls of the scenes and slides by using haptics devices [15] with the force feedback, head-mounted displays and healthcare virtual reality systems [18].
- Integrate stereoscopic effects into the presentation of softbody objects (under way) with another open-source plug-in project under development that implements OpenGL-based stereoscopic effects [4, 5, 18].
- While working on this and other integration efforts, take down and formalize software engineering requirements for systems like ours to simplify the future development and integration process of academic and open-source OpenGL and CG frameworks and systems for physical based animation and beyond [16] (in progress).
- Provide automatic loading and display of the softbody simulation source code on the slides with breakdown onto multi-page slides.
- Demonstrate different softbody shading techniques and shaders via the OpenGL slides. We already implemented the first draft version of a vendor-independent API for shader use within the softbody system and provided two implementations of that API – one that loads GLSL vertex and fragment shaders and the other that loads the cross-vendor assembly language for shaders.
- Demonstrate attachment of softbody objects to skeletal surfaces via the slides.

Acknowledgments

This work is partially funded by NSERC, FQRSC, and the Faculty of Engineering and Computer Science, Concordia University, Montreal, Canada.

References

[1] Autodesk. Maya. [digital], 2008–2009. `autodesk.com`.

[2] Peter Grogono. Concordia University Graphics Library (CUGL). [online], December 2005. `http://users.encs.concordia.ca/~grogono/Graphics/cugl.html`.

[3] Joaquim Jorge, Frank Hanisch, Frederico Figueiredo, and Rhonda Schauer. CG Educational Materials Source (CGEMS). [online], 2008–2009. `http://cgems.inesc.pt/`.

[4] Alison R. Loader, Serguei A. Mokhov, and Miao Song. Open Stereoscopic 3D Plugin Collection. SourceForge.net, 2008–2009. `http://sf.net/projects/stereo3d`, last viewed November 2009.

[5] Alison Reiko Loader. Making space. Master's thesis, Department of Design and Computation Arts, Concordia University, Montreal Canada, 2008.

[6] Serguei A. Mokhov and Miao Song. An OpenGL-based interface to 3D PowerPoint-like presentations of OpenGL projects. In *Proceedings of CISSE'08*, University of Bridgeport, CT, USA, December 2008. Springer. To appear.

[7] Serguei A. Mokhov and Miao Song. OpenGL project presentation slides interface and a case study. In *Proceedings of GRAPP'09*, pages 409–412, Lisboa, Portugal, February 2009. INSTICC.

[8] OpenGL Architecture Review Board. OpenGL. [online], 1998–2009. `http://www.opengl.org`.

[9] Paul Rademacher, Nigel Stewart, and Bill Baxter. GLUI – A GLUT-based user interface library, version 2.35. [online], 1999–2006. `http://glui.sourceforge.net/`.

[10] Randi J. Rost. *OpenGL Shading Language*. Pearson Education, Inc., February 2004. ISBN: 0-321-19789-5.

[11] Miao Song. Dynamic deformation of uniform elastic two-layer objects. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, August 2007.

[12] Miao Song and Peter Grogono. A framework for dynamic deformation of uniform elastic two-layer 2D and 3D objects in OpenGL. In *Proceedings of C3S2E'08*, pages 145–158, Montreal, Quebec, Canada, May 2008. ACM. ISBN 978-1-60558-101-9.

[13] Miao Song and Peter Grogono. An LOD control interface for an OpenGL-based softbody simulation framework. In *Proceedings of CISSE'08*, University of Bridgeport, CT, USA, December 2008. Springer. To appear.

[14] Miao Song and Peter Grogono. Application of advanced rendering and animation techniques for 3D games to softbody modeling and animation. In *Proceedings of C3S2E'09*, pages 89–100, Montreal, Quebec, Canada, May 2009. ACM.

[15] Miao Song and Peter Grogono. Are haptics-enabled interactive and tangible cinema, documentaries, 3D games, and specialist training applications our future? In *Proceedings of GRAPP'09*, pages 393–398, Lisboa, Portugal, February 2009. INSTICC.

[16] Miao Song, Serguei A. Mokhov, and Peter Grogono. Deriving software engineering requirements specification for computer graphics simulation systems through a case study. Submitted for publication to SERA'2010, December 2009.

[17] Miao Song, Serguei A. Mokhov, and Peter Grogono. Designing an interactive OpenGL slide-based presentation of the softbody simulation system for teaching and learning of computer graphics techniques. In *Proceedings of C3S2E'09*, pages 131–136, New York, NY, USA, May 2009. ACM.

[18] Miao Song, Serguei A. Mokhov, Alison R. Loader, and Maureen J. Simmonds. A stereoscopic OpenGL-based interactive plug-in interface for Maya and beyond. In *Proceedings of VRCAI'09*, New York, NY, USA, 2009. ACM. To appear.

[19] Jerry O. Talton. Teaching graphics with the OpenGL Shading Language. *ACM SIGCSE Bulletin archive*, 39(1), March 2007.

[20] Dana Tenneson, Anne Morgan Spalter, Julie Kumar, Ilya Medvedev, and Andries van Dam. The Graphics Teaching Tool (GTT). [online], Brown University, 2003 – 2008. `http://graphics.cs.brown.edu/research/gtt/`.

[21] Wikipedia. Procedural modeling — Wikipedia, The Free Encyclopedia. [online; accessed 28-November-2009], 2009. `http://en.wikipedia.org/w/index.php?title=Procedural_modeling&oldid=326319778`.

[22] Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner, and OpenGL Architecture Review Board. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley, 3 edition, October 1999. ISBN 0201604582.